Europäisches Patentamt | European Patent Office | Office européen des brevets

Bescheinigung Certificate Attestation

Die angehefteten Unterlagen stimmen mit der ursprünglich eingereichten Fassung der auf dem nächsten Blatt bezeichneten europäischen Patentanmeldung überein.

The attached documents are exact copies of the European patent application described on the following page, as originally filed.

Les documents fixés à cette attestation sont conformes à la version initialement déposée de la demande de brevet européen spécifiée à la page suivante.

Patentanmeldung Nr. Patent application No. Demande de brevet n°

01123052.1

Der Präsident des Europäischen Patentamts;
Im Auftrag

For the President of the European Patent Office

Le Président de l'Office européen des brevets
p.o.

R C van Dijk

DEN HAAG, DEN
THE HAGUE, 13/11/01

Europäisches Patentamt
European Patent Office
Office européen des brevets

Blatt 2 der Bescheinigung
Sheet 2 of the certificate
Page 2 de l'attestation

Anmeldung Nr.:
Application no.:
Demande n°: 01123052.1

Anmeldetag:
Date of filing:
Date de dépôt: 26/09/01

Anmelder:
Applicant(s):
Demandeur(s):
SAP Aktiengesellschaft
69190 Walldorf
GERMANY

Bezeichnung der Erfindung:
Title of the invention:
Titre de l'invention:
Communication between computers operating in different object-oriented run-time environnements

In Anspruch genommene Priorität(en) / Priority(ies) claimed / Priorité(s) revendiquée(s)

Staat: State: Pays:	Tag: Date: Date:	Aktenzeichen: File no. Numéro de dépôt:

Internationale Patentklassifikation:
International Patent classification:
Classification internationale des brevets:
/

Am Anmeldetag benannte Vertragstaaten:
Contracting states designated at date of filing: AT/BE/CH/CY/DE/DK/ES/FI/FR/GB/GR/IE/IT/LI/LU/MC/NL/PT/SE/TR
Etats contractants désignés lors du dépôt:

Bemerkungen:
Remarks:
Remarques:

COMMUNICATION BETWEEN COMPUTERS OPERATING IN DIFFERENT OBJECT-ORIENTED RUN-TIME ENVIRONMENTS

Field of the Invention

01 The present invention generally relates to data processing and, more particularly, relates to computer systems, computer programs, and methods to use properties and method of objects in computer business applications.

Background of the Invention

02 Computer applications often use object-oriented programming techniques. In the applications, objects are data structures with properties and methods. For convenience of explanation, the term "function" is used instead so that an object has properties and functions.

03 The objects are often arranged in hierarchy. Reading ("GET") and modifying ("SET") a property or executing a function ("INSTRUCT") requires identification (ID) of the object within the hierarchy, for example, by an object name. Technically, each object operates in a run-time environment, that means in a predefined technical format for the object ("object model"). For example, (a) the ID is a string with a predetermined number of characters and (b) an output value is an integer.

04 Multiple applications are often linked in a network so that using a single run-time environment is often impossible. When first and second computer applications operate in different first and second

run-time environments, they conveniently use similar objects with similar properties and functions.

05 However, similarity alone does not guarantee technical compatibility. Translation software between the applications is required as well. Linking the application by communication middleware is well known in the art. Middleware is commercially available, such as for example, COM, DCOM, and CORBA.

06 For example, a first run-time environment (provider) needs to communicate with an identified object in a second run-time environment (consumer). Such communication often involves multiple hierarchy levels of objects. Going through multiple hierarchy levels by both environments technically causes network load and delays.

07 There is an ongoing need to provide improved inter-run-time communication means so that some or all of the above disadvantages are mitigated.

Summary of the Invention

08 The present invention relates to a method for communication between a first computer operating in a first object-oriented run-time environment and a second computer operating in a second, different object-oriented run-time environment. The first run-time environment accesses a property or a function of an object, hereinafter "action", in the second run-time environment. The first computer sends a first message with object identification and action identification to the second computer and causes the second computer to perform the following steps: identifying the object in the second run-time environment according to the object identification;

verifying the existence of the action in the identified object in the second run-time environment; determining a representation of the action in the second run-time environment for the identified object; and executing the action by using the representation, thereby returning a second message to the first computer with object identification and response identification.

09 While identifying, interaction between the computers is not required; and execution is technically independent from the first run-time environment, since only the second run-time environment has to be used.

010 Preferably, step executing comprises: extracting a representation of a property that is identified by the action identification; converting the representation to a further representation for the first run-time environment; and returning the second message as result message to the first computer with object identification and response identification, the response identification with the representation for the first run-time environment.

011 Preferably, step executing comprises: converting a request identification that is part of the action identification to a representation for the second run-time environment; and inserting the representation into the second application.

012 Preferably, step executing comprises to return the second message as a confirmation message to the first computer.

013 Preferably step converting involves looking up in a table.

014 Preferably, wherein step executing comprises: converting function identification and parameter

identification of the action identification to function and parameter representations for the second run-time environment; performing a function that is identified by the action identification by using the function and parameter representations for the second run-time environment; converting parameters that result from executing into further representations for the first run-time environment; and returning the second message as a feedback message to the first computer with object identification and response identification, the response identification with the further representations.

015 The invention also relates to a computer program product used in a communication system of a first computer with a first object-oriented run-time environment and a second computer with a second, different object-oriented run-time environment, wherein the first computer sends a first message with object identification and action identification to the second computer, the computer program product embodied on a carrier and having computer code instructions to cause a processor of the second computer to interpret the first message, the instructions comprising: code for identifying an object in the second run-time environment according to the object identification; code for verifying the existence of the action in the identified object in the second run-time environment; code for determining a representation of the action in the second run-time environment for the identified object; and code for executing the action by using the representation and to return a second message to the first computer with object identification and response identification.

016 Preferably, the code for executing comprises: code for extracting a representation of a property that is identified by the action identification; code for converting the representation to a further representation for the first run-time environment; and code for returning the second message as a result message to the first computer with object identification and response identification, the response identification with the representation for the first run-time environment.

017 Preferably, the code for executing comprises: code for converting a request identification that is part of the action identification to a representation for the second run-time environment; and code for inserting the representation into the second application.

018 Preferably, the code for executing causes the processor to return the second message as a confirmation message.

019 Preferably, the code for converting comprises code to look up in a table.

020 The invention also relates to a computer communication system with a first computer operating in a first object-oriented run-time environment and a second computer operating in a second, different object-oriented run-time environment, wherein the first computer sends a first message with object identification and action identification to the second computer, the system characterized in that the second computer comprises:

021 a first module to identify an object in the second run-time environment according to the object identification;

022 a second module to verify the existence of the action in the identified object in the second run-time environment;

023 a third module to determine a representation of the action in the second run-time environment for the identified object; and

024 a fourth module to execute the action by using the representation, thereby returning a second message to the first computer with object identification and response identification.

025 Preferably, the fourth module executes any of the method steps executing.

Brief Description of the Drawings

026 FIG. 1 illustrates a simplified block diagram of a computer network system having a plurality of computers;

027 FIG. 2 illustrates a simplified block diagram of first and second computers with first and second applications, in first and second run-time environments, respectively, that communicate according to the present invention;

028 FIG. 3 illustrates messages from the first computer to the second computer;

029 FIG. 4 illustrates messages from the second computer to the first computer;

030 FIG. 5 illustrates an object in the second application on the second computer;

031 FIG. 6 illustrates a conversion table in the second application on the second computer;

032 FIG. 7 illustrates a flowchart diagram for a method of the present invention in general; and

033 FIG. 8 illustrates flowchart diagrams for the method of FIG. 7 for first, second and third embodiments.

Computer System in General

034 **FIG. 1** illustrates a simplified block diagram of a computer network system **999** having a plurality of computers **900**, **901**, **902** (or **90q**, with q=0...Q-1, Q any number).

035 Computers **900-902** are coupled via inter-computer network **990**. Computer **900** comprises processor **910**, memory **920**, bus **930**, and, optionally, input device **940** and output device **950** (I/O devices, user interface **960**). As illustrated, the invention is present by computer program product **100** (CPP), program carrier **970** and program signal **980**, collectively "program".

036 In respect to computer **900**, computer **901/902** is sometimes referred to as "remote computer", computer **901/902** is, for example, a server, a router, a peer device or other common network node, and typically comprises many or all of the elements described relative to computer **900**. Hence, elements **100** and **910-980** in computer **900** collectively illustrate also corresponding elements **10q** and **91q-98q** (shown for q=0) in computers **90q**.

037 Computer **900** is, for example, a conventional personal computer (PC), a desktop and hand-held device, a multiprocessor computer, a pen computer, a microprocessor-based or programmable consumer electronics, a minicomputer, a mainframe computer, a personal mobile computing device, a mobile phone, a

portable or stationary personal computer, a palmtop computer or the like.

038 Processor **910** is, for example, a central processing unit (CPU), a micro-controller unit (MCU), digital signal processor (DSP), or the like.

039 Memory **920** symbolizes elements that temporarily or permanently store data and instructions. Although memory **920** is conveniently illustrated as part of computer **900**, memory function can also be implemented in network **990**, in computers **901/902** and in processor **910** themselves (e.g., cache, register), or elsewhere. Memory **920** can be a read only memory (ROM), a random access memory (RAM), or a memory with other access options. Memory **920** is physically implemented by computer-readable media, such as, for example: (a) magnetic media, like a hard disk, a floppy disk, or other magnetic disk, a tape, a cassette tape; (b) optical media, like optical disk (CD-ROM, digital versatile disk - DVD); (c) semiconductor media, like DRAM, SRAM, EPROM, EEPROM, memory stick, or by any other media, like paper.

040 Optionally, memory **920** is distributed across different media. Portions of memory **920** can be removable or non-removable. For reading from media and for writing in media, computer **900** uses devices well known in the art such as, for example, disk drives, tape drives.

041 Memory **920** stores support modules such as, for example, a basic input output system (BIOS), an operating system (OS), a program library, a compiler, an interpreter, and a text- processing tool. Support modules are commercially available and can be installed on computer **900** by those of skill in the

art. For simplicity, these modules are not illustrated.

042 CPP **100** comprises program instructions and - optionally - data that cause processor **910** to execute method steps of the present invention. Method steps are explained with more detail below. In other words, CPP **100** defines the operation of computer **900** and its interaction in network system **999**. For example and without the intention to be limiting, CPP **100** can be available as source code in any programming language, and as object code ("binary code") in a compiled form. Persons of skill in the art can use CPP **100** in connection with any of the above support modules (e.g., compiler, interpreter, operating system).

043 Although CPP **100** is illustrated as being stored in memory **920**, CPP **100** can be located elsewhere. CPP **100** can also be embodied in carrier **970**.

044 Carrier **970** is illustrated outside computer **900**. For communicating CPP **100** to computer **900**, carrier **970** is conveniently inserted into input device **940**. Carrier **970** is implemented as any computer readable medium, such as a medium largely explained above (cf. memory **920**). Generally, carrier **970** is an article of manufacture comprising a computer readable medium having computer readable program code means embodied therein for executing the method of the present invention. Further, program signal **980** can also embody computer program **100**. Signal **980** travels on network **990** to computer **900**.

045 Having described CPP **100**, program carrier **970**, and program signal **980** in connection with computer **900** is convenient. Optionally, program carrier **971/972** (not shown) and program signal **981/982** embody computer program product (CPP) **101/102** to be executed by

processor **911/912** (not shown) in computers **901/902**, respectively.

046 Input device **940** symbolizes a device that provides data and instructions for processing by computer **900**. For example, device **940** is a keyboard, a pointing device (e.g., mouse, trackball, cursor direction keys), microphone, joystick, game pad, or scanner. Although the examples are devices with human interaction, device **940** can also operate without human interaction, such as, a wireless receiver (e.g., with satellite dish or terrestrial antenna), a sensor (e.g., a thermometer), a counter (e.g., goods counter in a factory). Input device **940** can serve to read carrier **970**.

047 Output device **950** symbolizes a device that presents instructions and data that have been processed. For example, a monitor or a display, (cathode ray tube (CRT), flat panel display, liquid crystal display (LCD), speaker, printer, plotter, vibration alert device. Similar as above, output device **950** communicates with the user, but it can also communicate with further computers.

048 Input device **940** and output device **950** can be combined to a single device; any device **940** and **950** can be provided optional.

049 Bus **930** and network **990** provide logical and physical connections by conveying instruction and data signals. While connections inside computer **900** are conveniently referred to as "bus **930**", connections between computers **900-902** are referred to as "network **990**". Optionally, network **990** comprises gateways being computers that specialize in data transmission and protocol conversion.

050 Devices **940** and **950** are coupled to computer **900** by bus **930** (as illustrated) or by network **990** (optional). While the signals inside computer **900** are mostly electrical signals, the signals in network are electrical, magnetic, optical or wireless (radio) signals.

051 Networking environments (as network **990**) are commonplace in offices, enterprise-wide computer networks, intranets and the Internet (i.e. world wide web). The physical distance between a remote computer and computer **900** is not important. Network **990** can be a wired or a wireless network. To name a few network implementations, network **990** is, for example, a local area network (LAN), a wide area network (WAN), a public switched telephone network (PSTN); a Integrated Services Digital Network (ISDN), an infra-red (IR) link, a radio link, like Universal Mobile Telecommunications System (UMTS), Global System for Mobile Communication (GSM), Code Division Multiple Access (CDMA), or satellite link.

052 Transmission protocols and data formats are known, for example, as transmission control protocol/internet protocol (TCP/IP), hyper text transfer protocol (HTTP), secure HTTP, wireless application protocol, unique resource locator (URL), a unique resource identifier (URI), hyper text markup language HTML, extensible markup language (XML), extensible hyper text markup language (XHTML), wireless application markup language (WML), Standard Generalized Markup Language (SGML) etc.

053 Interfaces coupled between the elements are also well known in the art. For simplicity, interfaces are not illustrated. An interface can be, for example, a serial port interface, a parallel port interface, a

game port, a universal serial bus (USB) interface, an internal or external modem, a video adapter, or a sound card.

054 Computer and program are closely related. As used hereinafter, phrases, such as "the computer provides" and "the program provides", are convenient abbreviation to express actions by a computer that is controlled by a program.

Detailed Description of the Invention

055 For convenience, a list of references is provided prior to the claims.

056 **FIG. 2** illustrates a simplified block diagram of first computer **901** and second computer **902** that communicate according to the present invention. Computers **901** and **902** perform applications **201** and **202**, in first and second run-time environments, respectively.
Application **201** operates in a first run-time environment that has a first object model; application **202** operates in a second, different run-time environment that has a second object model. Objects **A1**, **B1** and **C1** in application **201** correspond to objects **A2**, **B2** and **C2** in application **202**. **FIG. 2** illustrates this mapping by dotted lines. For explanation, distinguishing the letters A, B and C provides sufficient object identification. Indices 1 and 2 conveniently distinguish the computers. Object **A2** is explained with more detail in **FIG. 5** (reference **152**).

057 In operation, computers **901** and **902** preferably visualize their performance on screens **951** and **952** to

first and second users, respectively. For example, screen **951** on computer **901** shows representations obtained from computer **902** (cf. **FIG. 4**, **118**, **128**, **138**). Both users are mentioned here only for convenience of explanation; the present invention does not require them to be involved.

058 Network **990** (cf. **FIG. 1**) is the infrastructure to convey information between computers **901** and **902** in both directions. In both computers, first and second run-time environments cooperate with each other, preferably, by sending messages through network **990**. The messages are convenient for explanation, but not essential for the present invention. In the alternative, the communication is performed by computer middleware, such as COM or CORBA, well known in the art and not further illustrated.

059 Communicator **101** operates on computer **901** as a message generator and sends messages GET **111**, SET **121**, INSTRUCT **131** to computer **902** (arrows to the right). As explained in connection with **FIG. 3**, the messages carry object identification and action identification.

060 Interpreter **102** (i.e. CPP) operates on computer **902** as message interpreter according to a method of the present invention (cf. **FIG. 4**). Interpreter **102** sends messages RESULT **112**, CONFIRM **122**, FEEDBACK **132** and ERROR **192** to computer **901**. Es explained in connection with **FIG. 4**, the messages carry object ID and response ID. It is an advantage of the present invention that the response ID has representations of object properties and object functions suitable for the first run-time environment of computer **901**. As mentioned, the representations can be shown on screen.

061 There is no need to use all messages that are illustrated in **FIG. 2**. Preferably, the messages are exchanged as paired messages (the first message from computer **901**, the second messages from computer **902**), each pair for an embodiment of the present invention (GET/RESULT, SET/CONFIRM and INSTRUCT/FEEDBACK).

062 The first embodiment relates to the action of reading (GET) a property of an object (A) und uses messages GET **111** and RESULT **112**. The second embodiment relates to the action of modifying (SET) a property of the object (A) and uses messages SET **121** and CONFIRM **122**. The third embodiment relates to the action of executing (INSTRUCT) a function of the object (A) and uses messages INSTRUCT **131** and FEEDBACK **132**. Error message **192** from computer **902** to computer **902** is optionally used in all embodiments.

063 An exemplary scenario relates to the automotive industry. A customer ("first user") uses first application **201** to customize a car; a technician ("second user") in a factory uses second application **202** to response to the customer. Software objects correspond to hardware components; objects **A**, **B** and **C** correspond to cars ALPHA, BETA, and GAMMA, respectively (not shown). Applications **201** and **202** communicate with their users in first ° and second °° natural languages (e.g., German and Spanish). To conveniently describe the invention in a single language, symbols ° and °° distinguish first and second natural languages, respectively. For example, "green °" stands for the German word "grün" and "green °°" stands for the Spanish word "verde", both in the same meaning of green color. In a real implementation, these superscripts are not shown.

064 For convenience of explanation, the example scenario further assumes consecutive execution of the embodiments. The customer for ALPHA

- first, asks for the color ("GREEN ° "),
- second, changes the color ("to RED °"), and
- third, communicates with applications **201** and **202** to check volume ("LITER °") and price ("EURO") for a desired color type ("METALLIC °").

065 **FIG. 3** illustrates messages GET **111**, SET **121** and INSTRUCT **131** from computer **901** to computer **902** (cf. **FIG. 2**, arrows to the right). The messages comprise object identification (ID) **115**, **125** or **135** (e.g., object A) and action identification (ID) **116/117**, **126/127**, or **136/137**. For first or second embodiments, the action ID comprises property ID **116** or **126** and request ID **117** or **127**. For the third embodiment, the action ID comprises function ID **136** and parameter ID **137** (parameter representation).

066 **FIG. 4** illustrates messages RESULT **112**, CONFIRM **122**, and FEEDBACK **132** from computer **902** to computer **901** (cf. **FIG. 2**, arrows to the left). The messages comprise object ID **115**, **125** or **135** (e.g., object A, as in **FIG. 3**) and response ID **116/118** or **126/128** or **136/138**.

For first or second embodiments, the response ID comprises property ID **116** or **126** (as in **FIG. 3**) and property representation **118** or **128**. Property representation is that of first run-time environment in computer **901**.

For the third embodiment, the response ID comprises function ID **136** (as in **FIG. 3**) and parameter representation **138**. Function and parameter

representation are suitable to first run-time environment of computer **901**.

067 The messages in FIGS. 3-4 are now explained in connection with the embodiments. So far, error messages **192** are neglected.

068 In the first embodiment, the customer needs to know the color of the car ALPHA. Application **201** uses object **A1** and communicator **101** to send GET **111** with object ID **115** "A", property ID **116** "COLOR", and request ID **117** "GET COLOR" (action).

069 At computer **902**, interpreter **102** executes a method (cf. **410** in **FIG. 8**) in combination with corresponding object **A2** (of application **202**) and returns RESULT **112** with corresponding object ID **115** and property ID **116**, but with property representation **118** "COLOR GREEN °" (response). Representation **118** is in the suitable format for application **201**. Application **201** now indicates "Green ° to the customer (e.g. on screen **951**).

070 The role of user of application **202** is optional. When executing the method, interpreter **102** converts representations to and from the different run-time environments. It is convenient (but not necessary) that application **202** shows the color to the technician in the second language °° or invites the technician to input the color in this second language °°.

071 In the second embodiment, the customer wishes to set the color of the car ALPHA to "red". Application **201** uses object **A1** and communicator **101** to send SET **121** to application **202** with object ID **125** "A", property ID **126** "COLOR", and request ID **127** "SET TO RED °°.

072 At computer **902**, interpreter **102** executes a method (cf. **420** in **FIG. 8**) in combination with corresponding object **A2** (of application **202**) and returns CONFIRM **122** to application **201** with corresponding object ID **125** and property ID **126**, but with property representation **128** "YES, SET TO RED °".

073 In the third embodiment, the customer communicates with applications **201** and **202** to check price and volume for a desired color type. Application **201** again uses object **A1** and communicator **101** to send INSTRUCT **131** to application **202** with function ID **136** "CALCULATE" and parameter ID **137** "COLOR TYPE ° = METALLIC °", "VOLUME ° = X", "PRICE ° = Y". The parameters are input parameter (COLOR TYPE) and output parameters (VOLUME, PRICE).

074 At computer **902**, interpreter **102** executes the method in combination with corresponding object **A2**. Interpreter identifies object **A2**, verifies the parameters of **A2**, determines parameter representations, converts representations to the second-run time environment, triggers application **202** to execute the calculations, converts the calculation results X °° and Y °° to representations for the first run-time environment, and returns FEEDBACK **132** that indicates the output parameters (X °° and Y °°) as numeric values.

075 Before explaining the methods, details for the software at computer **902** are given.

076 **FIG. 5** illustrates object **152** (**A2**) in application **202** of second computer **902**. Object **A2** has property COLOR. COLOR has a plurality of property representations **162**

(RED °°, YELLOW °°, and GREEN °°) in the second run-time environment. In the example of **FIG. 5**, COLOR is represented by strings in the second natural language (°°). One representation of COLOR is being selected (e.g. GREEN °°). For the third embodiment, object **A2** also has function **172** CALCULATE with parameters COLOR TYPE, VOLUME, and PRICE.

077 **FIG. 6** illustrates conversion table **182** (mapping table) in application **202** (optionally in interpreter 102) on computer **902**. Due to different run-time environments in computers **901** and **902**, properties and functions of corresponding objects are represented differently. Table **182** illustrates property representations **161** and **162**. According to the present invention, interpreter **102** on computer **902** converts representations **161** from the format of the first run-time environment (RTE) to representations **162** (cf. **FIG. 5**) in the format of the second run-time environment and vice versa. As an example, table **182** converts string representations in the first language (°) of the property COLOR to a string representation in the second language (°°). Again, using natural languages is convenient for explanation; in practice there is great variety of conversion between different object models, such as integer-to-real number conversion; string-to-string conversion; one-byte character representation (e.g., ASCII) to double-byte representation (e.g., unicode), etc.

078 Persons of skill in the art can provide further property conversion means, for example: string-to-integer, integer-to-real, 4-byte-integer to 8-byte-integer, etc. and vice versa. Table **182** illustrates property representations only; function conversion

including parameter conversion is likewise to the property conversion, persons of skill in the art can accomplish this without the need of further explanation herein.

079 **FIG. 7** illustrates flowchart diagrams for method **400** of the present invention, with method **400** being a generalization for all embodiments.

080 **FIG. 8** illustrates flowchart diagrams for the embodiments: first method **410**, second method **420** and third method **430**.

081 Generally, method **400** relates to communication between first computer **901** operating in a first object-oriented run-time environment and second computer **902** operating in a second, different object-oriented run-time environment, wherein computer **901** sends message (GET **111**, SET **121**, or INSTRUCT **131**) with object ID **115**, **125**, **135** and action ID **116/117**, **126/127** or **136/137** to second computer **902** and causes second computer **902** to perform the method steps identify **401**, verify **402**, determine **403**, and execute **405**, **404**, **406**, **407**. The method steps are performed by interpreter **102** in combination with application **202** after computer **902** has received message **111**, **121**, or **131**.

082 In step identifying **401** an object in the second run-time environment according to object ID **115**, **125**, **135**, interpreter **102** identifies the corresponding object in application **202** by evaluating the object ID of the message. Step **401** is likewise for all embodiments, i.e. steps **411**, **421**, and **431**. In the

example scenario, with object ID "A", the object is **A2** (cf. **FIG. 5**).

083 In step verifying **402**, interpreter **102** verifies the existence of the action in the identified object in the second run-time environment. In the first and second embodiments, steps **412** and **422**, interpreter evaluates action identifier **116** and **126** and determines existence or non-existence of PROPERTY COLOR. In case of non-existence, interpreter **102** issues error message **192** (cf. **FIG. 2**). In the third embodiment, step **432**, interpreter evaluates action identifier **136** and determines existence or non-existence of function CALCULATE.

084 In step determining **403** representation **162** of action in the second run-time environment for the identified object, interpreter **102** operates for each embodiment slightly different. In the first and second embodiments, steps **413** and **423**, interpreter **102** determines that representations **162** are in string format (cf. **FIG. 5**, °°). In the third embodiment, step **433**, interpreter determines that the function has representation **172** (cf. **FIG. 5**).

085 In steps executing **404**, **405**, **406** and **407**, interpreter **102** executes the action by using representation **162** (or **172**). Interpreter **102** thereby operates for each embodiment in a slightly different manner. Common is obtaining the requested information (EXTRACT **414**, PERFORM **435**), converting formats (CONVERT **415**, CONVERT **424**, CONVERT **436**) and returning messages (RETURN **417**, **427**, **437**). Messages **112**, **122**, **132** to computer **901** comprise object ID **115**, **125**, **135** and response ID **116/118**, **126/128**, **136/138**. The following

description now conveniently splits into first, second and third embodiments.

086 **First Embodiment**

087 In step extracting **414**, interpreter **102** extracts representation **162** of property COLOR that is identified by action ID **116** (in GET **111**). So far representation **162** has the format for the second run-time environment (e.g., GREEN °°).

088 In step converting **415**, interpreter **102** converts representation **162** (GREEN °°) to a further representation (GREEN °, **161**) for the first run-time environment, for example, by reading from table **182** (cf. **FIG. 5**).

089 In step returning **417**, interpreter **102** return message RESULT **112** to the first computer with object ID **115** and response ID **116/118** (cf. **FIG. 4**). Response ID **116/118** comprises representations for the first run-time environment (GREEN °) as property representation **118**. In other words, interpreter **102** includes the extracted property into message **112** in a format suitable for the run-time of message-receiving computer **901**.

090 **Second Embodiment**

091 As property data is communicated to computer **902**, extracting and converting change the order, and inserting replaces extracting.

092 In step converting **424**, interpreter **102** converts request ID **127** (part of the action ID **126/127**) to representation **162** for the second run-time environment. For example, interpreter looking up in table **182** (cf. **FIG. 3**) to convert RED ° to RED °°.

093 In step inserting **425**, interpreter **102** inserts representation **162** into application **202**. In the example scenario, RED ° is stored in application such that car ALPHA will actually be painted red.

094 In step returning **427**, interpreter **102** sends message **122** to computer **901**. As explained above, CONFIRM **122** has object ID **125** and response ID **126/128** with property representation **128**.

095 **Third Embodiment**

096 Executing comprises converting **434**, performing **435**, converting **436**, and returning **437** steps. Function parameters are treated similar as properties.

097 In step converting **434**, interpreter **102** converts function ID **136** and parameter ID **137** of action ID **136/137** to function and parameter representations **172** (cf. **FIG. 5**) for the second run-time environment. For example, the function CALCULATE with its parameters COLOR TYPE (e.g. METALLIC), VOLUME (e.g., X) and PRICE (e.g., Y) is converted.

098 In step performing **435**, interpreter **102** causes the identified object of application **202** to actually provide the output parameters. The object uses function and parameter representations **172** for the second run-time environment. In the example, object **A2** calculates VOLUME by multiplying the surface area of the car (e.g., square meters) with an area specific volume (e.g., liters per square meter), object **A2** also calculates PRICE my multiplying VOLUME with a specific price for METALLIC (e.g., currency per liters). The parameter representations are still in the second run-time environment.

099 In step converting **436**, interpreter **102** converts the parameters (that result from performing step **435**)

into further representations **138** for the first run-time environment. For example, interpreter **102** converts VOLUME into a numeric representation suitable for the first run-time environment (e.g., integer to real number conversion; optionally conversion of physical units); interpreter **102** converts PRICE into a numeric representation suitable for the first run-time environment, conveniently with currency conversion.

0100 In step returning **437**, interpreter **102** returns feedback message **132** to computer **901** with object ID **135** and response ID **136/138**. Response ID **136/138** comprises the further representations for the first run-time environment.

0101 Optionally, function CALCULATE uses property COLOR (cf. first and second embodiments). This is convenient, for example, when COLOR influences output parameters such as PRICE.

List of Reference Numbers

Reference	Element	Embodiment
101	communicator	
102	interpreter (CPP)	
111	GET message	first
112	RESULT message	first
115	object ID	first
116	property ID	first
116/117	action ID	first
116/118	response ID	first
117	request ID	first
118	property representation	first
121	SET message	second
122	CONFIRM message	second
125	object ID	second
126	property ID	second
126/127	action ID	second
126/128	response ID	second
127	request ID	second
128	property representation	second
131	INSTRUCT message	third
132	FEEDBACK message	third
135	object ID	third
136	function ID	third
136/137	action ID	third
136/138	response ID	third
137	parameter ID	third
138	parameter representation	third
152	object A2	
161	object representation, first RTE	
162	object representation,	

Reference	Element	Embodiment
	second RTE	
172	function and parameter representation	third
182	conversion table	
192	ERROR message	all
201	application	
202	application	
400	method and steps	
401-407	method steps	all
41x	method and steps	first
42x	method and steps	second
43x	method and steps	third
901	computer	
902	computer	
951, 952	screens	
990	network	
A1, B1, C1	objects	
A2, B2, C2	objects	
ID	identification	
RTE	run-time environment	
xx1, xx2	indices to distinguish computers	

Claims

1. A method (400, 410, 420, 430) for communication between a first computer (901) operating in a first object-oriented run-time environment and a second computer (902) operating in a second, different object-oriented run-time environment, wherein the first computer (901) sends a first message (GET, SET, INSTRUCT) with object identification (115, 125, 135) and action identification (116/117, 126/127, 136/137) to the second computer (902) and causes the second computer (902) to perform the following steps:

   - identifying (401) an object (A2) in the second run-time environment according to the object identification (115, 125, 135);
   - verifying (402) the existence of the action (116/126/136) in the identified object (A2) in the second run-time environment;
   - determining (403) a representation (162/172) of the action (116/126/136) in the second run-time environment for the identified object (A2); and
   - executing (404, 405, 406, 407) the action by using the representation (162/172), thereby returning a second message (112, 122, 132) to the first computer (901) with object identification (115, 125, 135) and response identification (116/118, 126/128, 136/138).

2. The method (410) of claim 1, wherein step executing comprises:

- extracting (414) a representation (162) of a property (COLOR) that is identified by the action identification (116);
- converting (415) the representation (GREEN °°, 162) to a further representation (GREEN °, 161) for the first run-time environment; and
- returning (417) the second message as result message (112) to the first computer with object identification (115) and response identification (116, 118), the response identification (116, 118) with the representation for the first run-time environment (GREEN °).

3. The method (420) of claim 1, wherein step executing comprises:

- converting (424) a request identification (127) that is part of the action identification (126/127) to a representation (162, RED °°) for the second run-time environment; and
- inserting (425) the representation (162, RED °) into the second application (202).

4. The method (420) of claim 3, wherein step executing comprises to return (427) the second message as a confirmation message (122) to the first computer.

5. The method (420) of claim 3, wherein step converting (424) involves looking up in a table (182 RED ° to RED °°).

6. The method (430) of claim 3, wherein step executing comprises:

converting (434) function identification and parameter identification of the action identification (136/137) to function and parameter representations (172) for the second run-time environment;

performing (435) a function that is identified by the action identification by using the function and parameter representations (172) for the second run-time environment;

converting (436) parameters that result from executing into further representations (136, 138) for the first run-time environment; and

returning (437) the second message as a feedback message (132) to the first computer with object identification (135) and response identification (136, 138), the response identification (136, 138) with the further representations.

7. A computer program product (102) used in a communication system of a first computer (901) with a first object-oriented run-time environment and a second computer (902) with a second, different object-oriented run-time environment, wherein the first computer (901) sends a first message (GET, SET, INSTRUCT) with object identification (115, 125, 135) and action identification (116/117, 126/127, 136/137) to the second computer (902), the computer program product (102) embodied on a carrier (972) and having computer code instructions to cause a processor (922) of the second computer to interpret the first message, the instructions comprising:

code for identifying (401) an object (A2) in the second run-time environment according to the object identification (115, 125, 135);

code for verifying (402) the existence of the action (116/126/136) in the identified object (A2) in the second run-time environment;

code for determining (403) a representation (162/172) of the action (116/126/136) in the second run-time environment for the identified object (A2); and

code for executing (404, 405, 406, 407) the action by using the representation (162/172) and to return a second message (112, 122, 132) to the first computer (901) with object identification (115, 125, 135) and response identification (116/118, 126/128, 136/138).

8. The computer program product (102) of claim 7, wherein the code for executing comprises:

code for extracting (414) a representation (162) of a property (COLOR) that is identified by the action identification (116);

code for converting (415) the representation (GREEN °°, 162) to a further representation (GREEN °, 161) for the first run-time environment; and

code for returning (417) the second message as a result message (112) to the first computer with object identification (115) and response identification (116, 118), the response identification (116, 118) with the representation for the first run-time environment (GREEN °).

9. The computer program product (102) of claim 7, wherein the code for executing comprises:

code for converting (424) a request identification (127) that is part of the action identification (126/127) to a representation (162, RED °°) for the second run-time environment; and

code for inserting (425) the representation (162, RED °) into the second application (202).

10. The computer program product (102) of claim 9, wherein the code for executing causes the processor to return (427) the second message as a confirmation message (122).

11. The computer program product (102) of claim 9, wherein the code for converting (424) comprises code to look up in a table (182 RED ° to RED °°).

12. The computer program product (102) of claim 9, the code for executing comprises:

code for converting (434) function identification and parameter identification of the action identification (136/137) to function and parameter representations (172) for the second run-time environment;

code for performing (435) a function that is identified by the action identification by using the function and parameter representations (172) for the second run-time environment;

code for converting (436) parameters that result from executing into further representations (136, 138) for the first run-time environment; and

code for returning (437) the second message as a feedback message (132) to the first computer with object identification (135) and response identification (136, 138), the response identification (136, 138) with the further representations.

13. A computer communication system with a first computer (901) operating in a first object-oriented run-time environment and a second computer (902) operating in a second, different object-oriented run-time environment, wherein the first computer (901) sends a first message (GET, SET, INSTRUCT) with object identification (115, 125, 135) and action identification (116/117, 126/127, 136/137) to the second computer (902), the system characterized in that the second computer (902) comprises:

a first module to identify (401) an object (A2) in the second run-time environment according to the object identification (115, 125, 135);

a second module to verify (402) the existence of the action (116/126/136) in the identified object (A2) in the second run-time environment;

a third module to determine (403) a representation (162/172) of the action (116/126/136) in the second run-time environment for the identified object (A2); and

a fourth module to execute (404, 405, 406, 407) the action by using the representation (162/172), thereby returning a second message (112, 122, 132) to the first computer (901) with object identification (115, 125, 135) and response identification (116/118, 126/128, 136/138).

14. The computer system of claim 13, wherein the fourth module executes any of the steps in method claims 2 to 6.

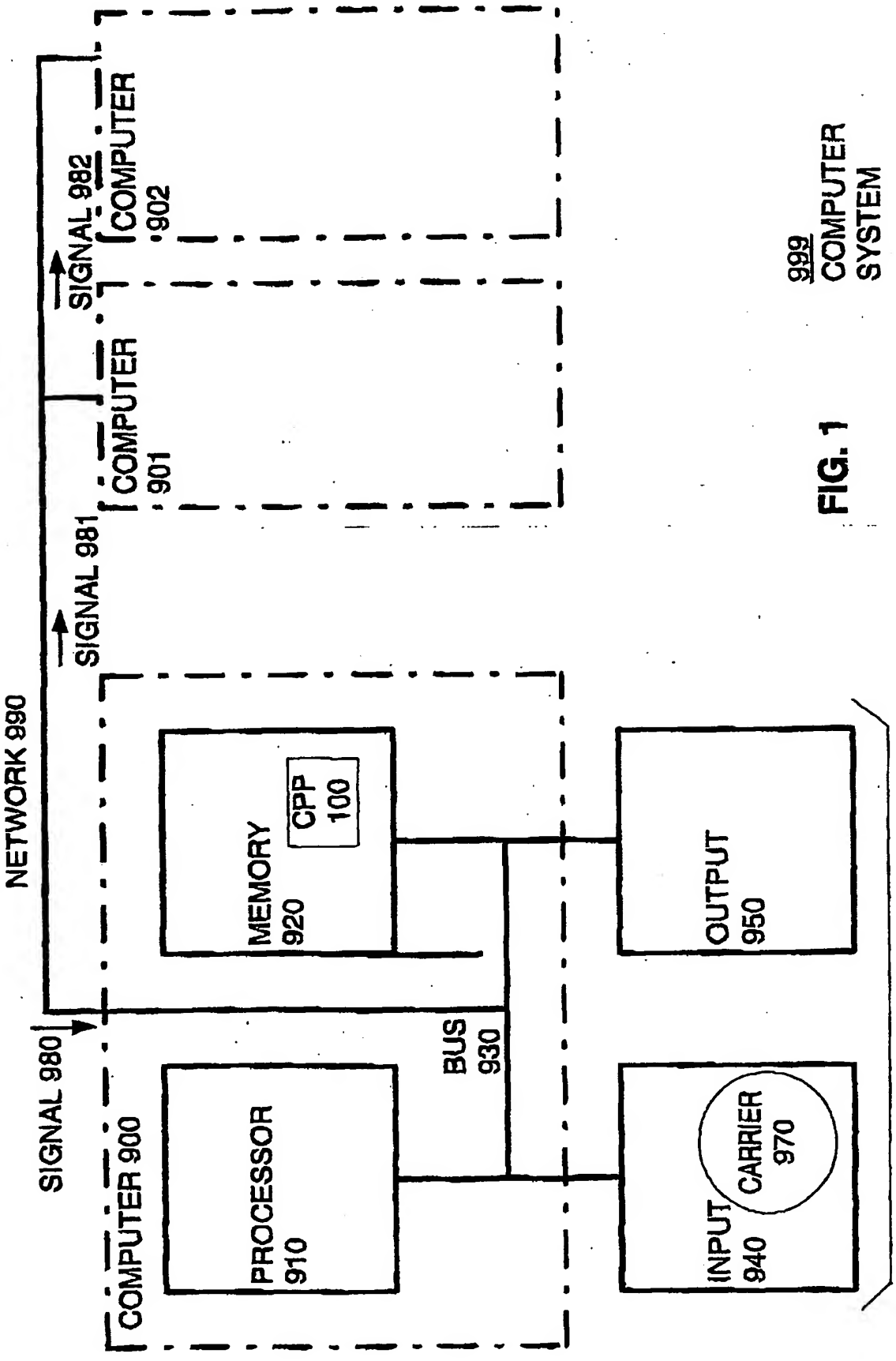

999
COMPUTER SYSTEM
NETWORK 990
COMPUTER 900
COMPUTER 901
COMPUTER 902
SIGNAL 980
SIGNAL 981
SIGNAL 982
PROCESSOR 910
MEMORY 920
CPP 100
BUS 930
INPUT 940
CARRIER 970
OUTPUT 950
USER INTERFACE 960

FIG. 1

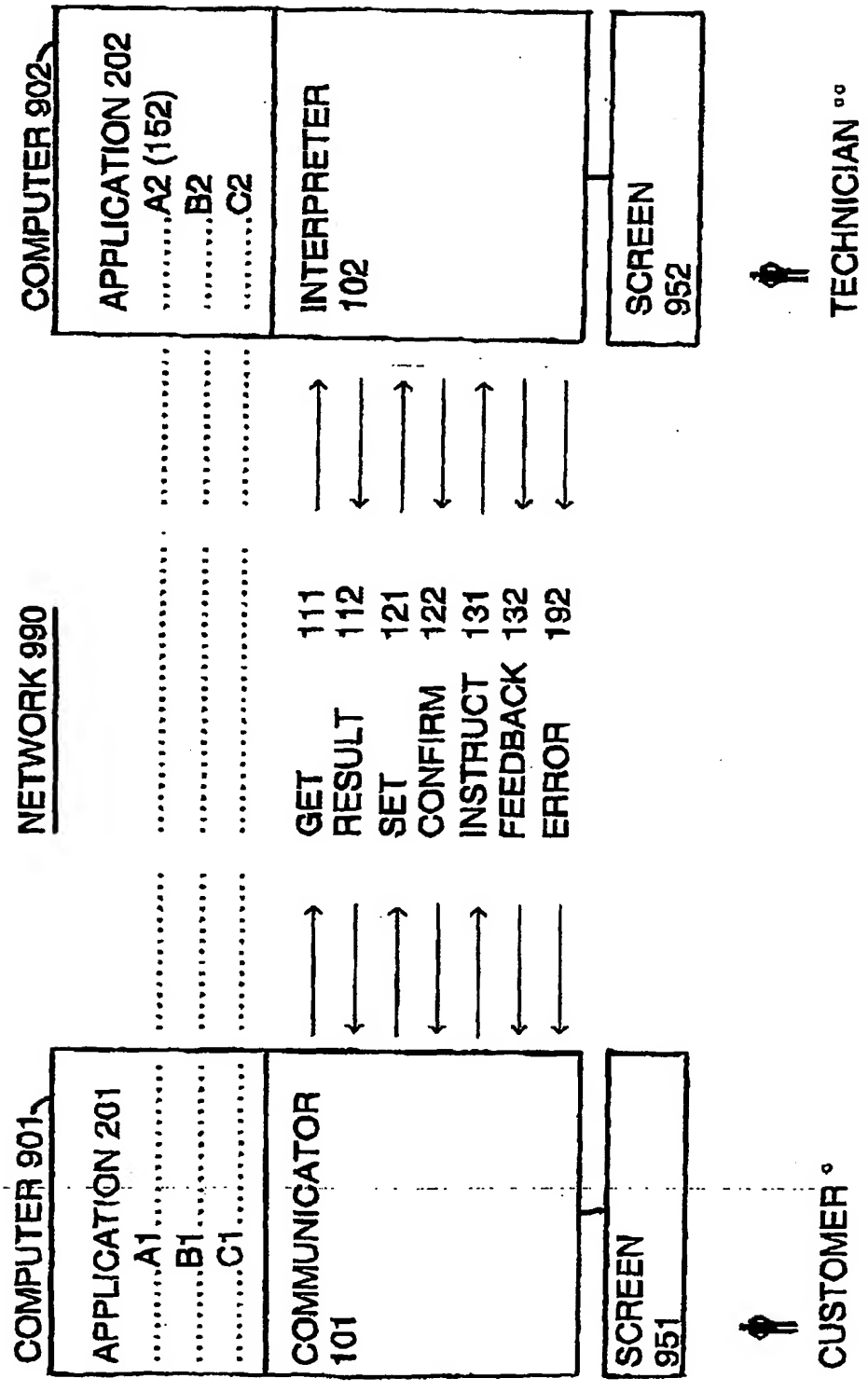

COMPUTER 901
APPLICATION 201
A1
B1
C1
COMMUNICATOR 101
SCREEN 951
CUSTOMER °
NETWORK 990
GET 111
RESULT 112
SET 121
CONFIRM 122
INSTRUCT 131
FEEDBACK 132
ERROR 192
COMPUTER 902
APPLICATION 202
A2 (152)
B2
C2
INTERPRETER 102
SCREEN 952
TECHNICIAN °°

FIG. 2

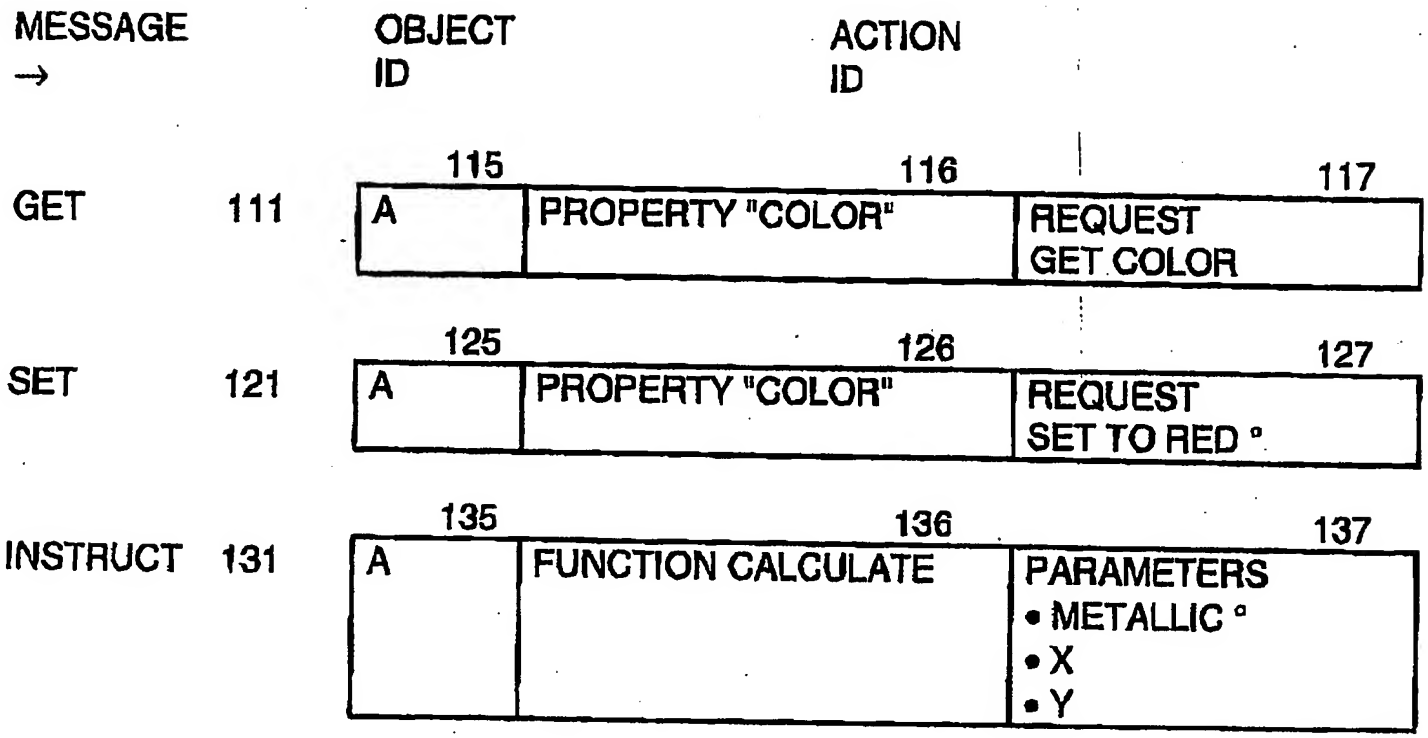

MESSAGE
→
OBJECT
ID
ACTION
ID
GET
111
115
116
117
A
PROPERTY "COLOR"
REQUEST
GET COLOR
SET
121
125
126
127
A
PROPERTY "COLOR"
REQUEST
SET TO RED °
INSTRUCT
131
135
136
137
A
FUNCTION CALCULATE
PARAMETERS
• METALLIC °
• X
• Y

FIG. 3

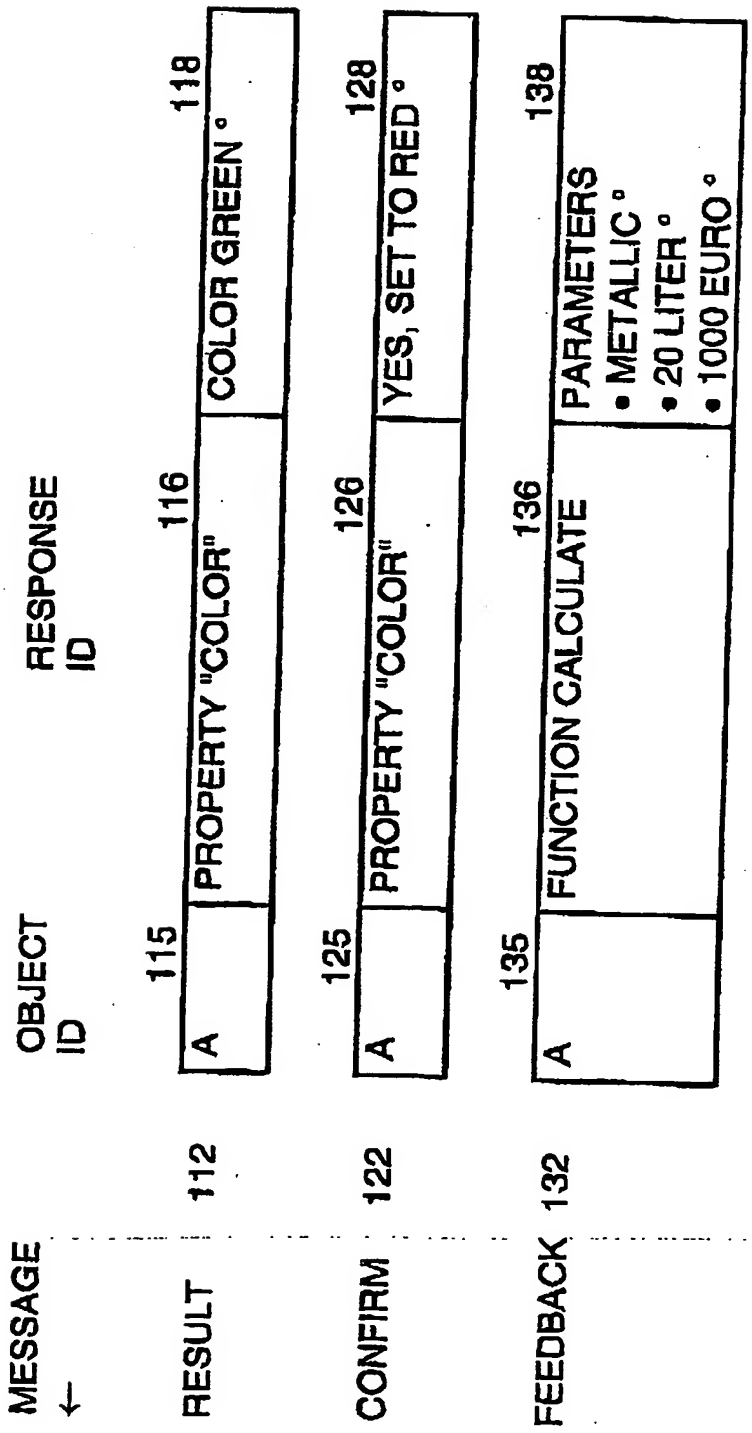

MESSAGE
OBJECT ID
RESPONSE ID
RESULT
112
115
A
116
PROPERTY "COLOR"
118
COLOR GREEN °
CONFIRM
122
125
A
126
PROPERTY "COLOR"
128
YES, SET TO RED °
FEEDBACK
132
135
A
136
FUNCTION CALCULATE
138
PARAMETERS
• METALLIC °
• 20 LITER °
• 1000 EURO °

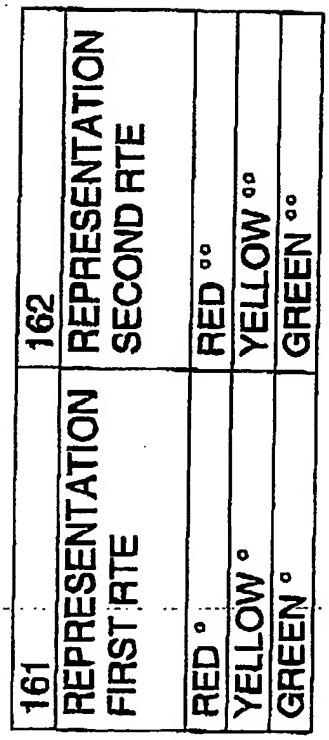
FIG. 4

OBJECT A2 152

PROPERTY 162	FUNCTION / PARAMETER 172
RED °°	CALCULATE • COLOR TYPE °° • VOLUME °° • PRICE °°
YELLOW °°	
GREEN °°	

FIG. 5

FIG. 6


161
REPRESENTATION FIRST RTE
162
REPRESENTATION SECOND RTE
RED °
RED °°
YELLOW °
YELLOW °°
GREEN °
GREEN °°
182

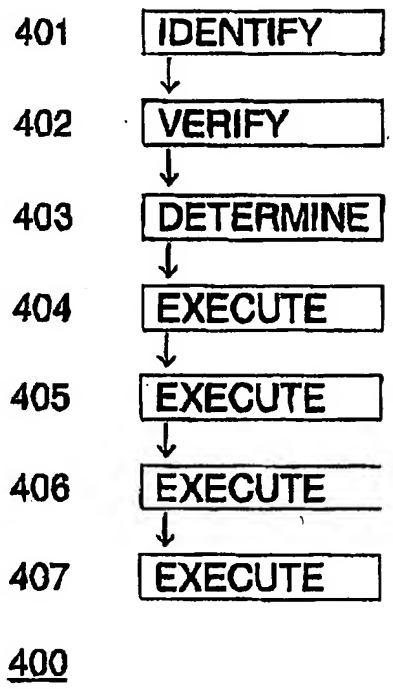

401
IDENTIFY
402
VERIFY
403
DETERMINE
404
EXECUTE
405
EXECUTE
406
EXECUTE
407
EXECUTE
400

FIG. 7

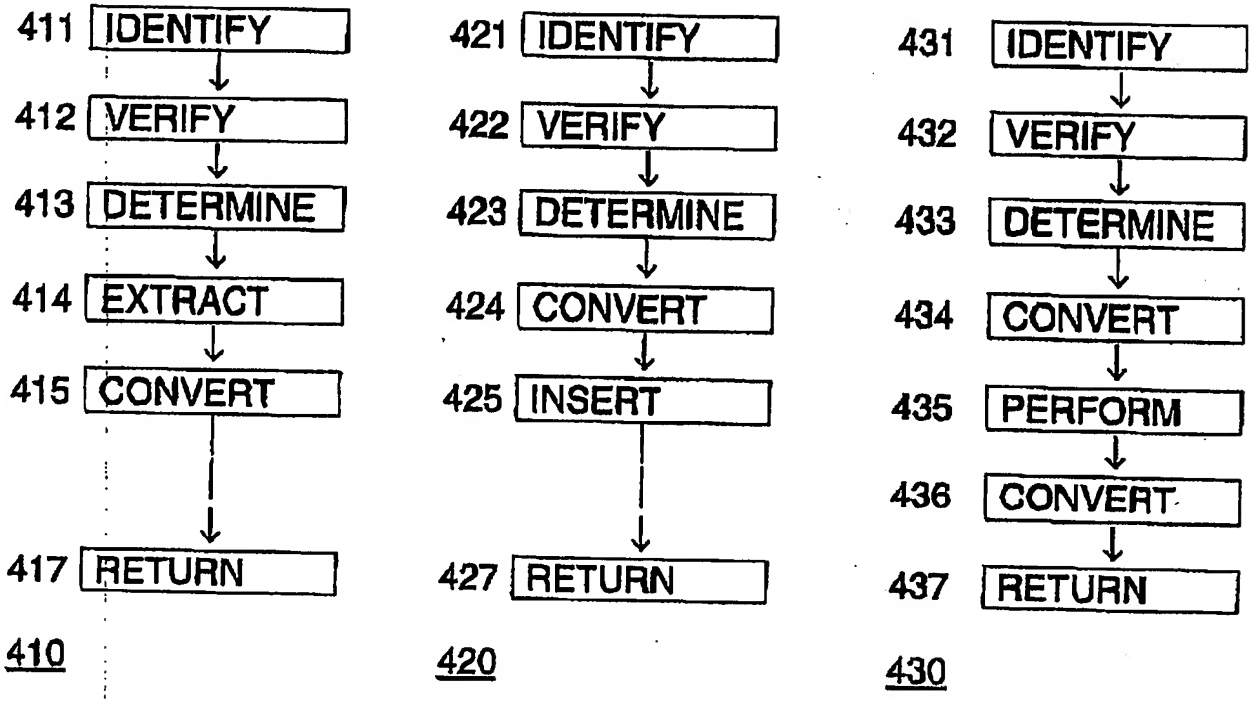

411
IDENTIFY
412
VERIFY
413
DETERMINE
414
EXTRACT
415
CONVERT
417
RETURN
410
421
IDENTIFY
422
VERIFY
423
DETERMINE
424
CONVERT
425
INSERT
427
RETURN
420
431
IDENTIFY
432
VERIFY
433
DETERMINE
434
CONVERT
435
PERFORM
436
CONVERT
437
RETURN
430

FIG. 8

COMMUNICATION BETWEEN COMPUTERS OPERATING IN DIFFERENT OBJECT-ORIENTED RUN-TIME ENVIRONMENTS

Abstract of the Disclosure

First and second computers (901, 902) operate in first and second object-oriented run-time environments with first and second applications (201, 202) and corresponding objects (A1/A2, B1/B2, C1/C2). Initiated by a communicator (101) and supported by a network (990), the first computer (901) sends a message (GET 111, SET 121, INSTRUCT 131) with object and action identification to the second computer (902). The second computer (902) uses an interpreter (102) to identify an object (A2, 152) in the second run-time environment according to the object identification. The interpreter (102) further verifies the existence of the identified action for the object (A2, 152) in the second run-time environment and determines (403) a representation of the action in the second run-time environment for the identified object (A2, 156). The interpreter (102) than executes the action by using the representation. Thereby, the interpreter (102) calls properties and methods of the identified object (A2, 156).

COMMUNICATION BETWEEN COMPUTERS OPERATING IN DIFFERENT OBJECT-ORIENTED RUN-TIME ENVIRONMENTS

Field of the Invention

01 The present invention generally relates to data processing and, more particularly, relates to computer systems, computer programs, and methods to use properties and method of objects in computer business applications.

Background of the Invention

02 Computer applications often use object-oriented programming techniques. In the applications, objects are data structures with properties and methods. For convenience of explanation, the term "function" is used instead so that an object has properties and functions.

03 The objects are often arranged in hierarchy. Reading ("GET") and modifying ("SET") a property or executing a function ("INSTRUCT") requires identification (ID) of the object within the hierarchy, for example, by an object name. Technically, each object operates in a run-time environment, that means in a predefined technical format for the object ("object model"). For example, (a) the ID is a string with a predetermined number of characters and (b) an output value is an integer.

04 Multiple applications are often linked in a network so that using a single run-time environment is often impossible. When first and second computer applications operate in different first and second

run-time environments, they conveniently use similar objects with similar properties and functions.

05 However, similarity alone does not guarantee technical compatibility. Translation software between the applications is required as well. Linking the application by communication middleware is well known in the art. Middleware is commercially available, such as for example, COM, DCOM, and CORBA.

06 For example, a first run-time environment (provider) needs to communicate with an identified object in a second run-time environment (consumer). Such communication often involves multiple hierarchy levels of objects. Going through multiple hierarchy levels by both environments technically causes network load and delays.

07 There is an ongoing need to provide improved inter-run-time communication means so that some or all of the above disadvantages are mitigated.

Summary of the Invention

08 The present invention relates to a method for communication between a first computer operating in a first object-oriented run-time environment and a second computer operating in a second, different object-oriented run-time environment. The first run-time environment accesses a property or a function of an object, hereinafter "action", in the second run-time environment. The first computer sends a first message with object identification and action identification to the second computer and causes the second computer to perform the following steps: identifying the object in the second run-time environment according to the object identification;

verifying the existence of the action in the identified object in the second run-time environment; determining a representation of the action in the second run-time environment for the identified object; and executing the action by using the representation, thereby returning a second message to the first computer with object identification and response identification.

009 While identifying, interaction between the computers is not required; and execution is technically independent from the first run-time environment, since only the second run-time environment has to be used.

010 Preferably, step executing comprises: extracting a representation of a property that is identified by the action identification; converting the representation to a further representation for the first run-time environment; and returning the second message as result message to the first computer with object identification and response identification, the response identification with the representation for the first run-time environment.

011 Preferably, step executing comprises: converting a request identification that is part of the action identification to a representation for the second run-time environment; and inserting the representation into the second application.

012 Preferably, step executing comprises to return the second message as a confirmation message to the first computer.

013 Preferably step converting involves looking up in a table.

014 Preferably, wherein step executing comprises: converting function identification and parameter

identification of the action identification to function and parameter representations for the second run-time environment; performing a function that is identified by the action identification by using the function and parameter representations for the second run-time environment; converting parameters that result from executing into further representations for the first run-time environment; and returning the second message as a feedback message to the first computer with object identification and response identification, the response identification with the further representations.

015 The invention also relates to a computer program product used in a communication system of a first computer with a first object-oriented run-time environment and a second computer with a second, different object-oriented run-time environment, wherein the first computer sends a first message with object identification and action identification to the second computer, the computer program product embodied on a carrier and having computer code instructions to cause a processor of the second computer to interpret the first message, the instructions comprising: code for identifying an object in the second run-time environment according to the object identification; code for verifying the existence of the action in the identified object in the second run-time environment; code for determining a representation of the action in the second run-time environment for the identified object; and code for executing the action by using the representation and to return a second message to the first computer with object identification and response identification.

016 Preferably, the code for executing comprises: code for extracting a representation of a property that is identified by the action identification; code for converting the representation to a further representation for the first run-time environment; and code for returning the second message as a result message to the first computer with object identification and response identification, the response identification with the representation for the first run-time environment.

017 Preferably, the code for executing comprises: code for converting a request identification that is part of the action identification to a representation for the second run-time environment; and code for inserting the representation into the second application.

018 Preferably, the code for executing causes the processor to return the second message as a confirmation message.

019 Preferably, the code for converting comprises code to look up in a table.

020 The invention also relates to a computer communication system with a first computer operating in a first object-oriented run-time environment and a second computer operating in a second, different object-oriented run-time environment, wherein the first computer sends a first message with object identification and action identification to the second computer, the system characterized in that the second computer comprises:

021 a first module to identify an object in the second run-time environment according to the object identification;

022 a second module to verify the existence of the action in the identified object in the second run-time environment;

023 a third module to determine a representation of the action in the second run-time environment for the identified object; and

024 a fourth module to execute the action by using the representation, thereby returning a second message to the first computer with object identification and response identification.

025 Preferably, the fourth module executes any of the method steps executing.

Brief Description of the Drawings

026 FIG. 1 illustrates a simplified block diagram of a computer network system having a plurality of computers;

027 FIG. 2 illustrates a simplified block diagram of first and second computers with first and second applications, in first and second run-time environments, respectively, that communicate according to the present invention;

028 FIG. 3 illustrates messages from the first computer to the second computer;

029 FIG. 4 illustrates messages from the second computer to the first computer;

030 FIG. 5 illustrates an object in the second application on the second computer;

031 FIG. 6 illustrates a conversion table in the second application on the second computer;

032 FIG. 7 illustrates a flowchart diagram for a method of the present invention in general; and

033 FIG. 8 illustrates flowchart diagrams for the method of FIG. 7 for first, second and third embodiments.

Computer System in General

034 **FIG. 1** illustrates a simplified block diagram of a computer network system **999** having a plurality of computers **900**, **901**, **902** (or **90q**, with q=0...Q-1, Q any number).

035 Computers **900-902** are coupled via inter-computer network **990**. Computer **900** comprises processor **910**, memory **920**, bus **930**, and, optionally, input device **940** and output device **950** (I/O devices, user interface **960**). As illustrated, the invention is present by computer program product **100** (CPP), program carrier **970** and program signal **980**, collectively "program".

036 In respect to computer **900**, computer **901/902** is sometimes referred to as "remote computer", computer **901/902** is, for example, a server, a router, a peer device or other common network node, and typically comprises many or all of the elements described relative to computer **900**. Hence, elements **100** and **910-980** in computer **900** collectively illustrate also corresponding elements **10q** and **91q-98q** (shown for q=0) in computers **90q**.

037 Computer **900** is, for example, a conventional personal computer (PC), a desktop and hand-held device, a multiprocessor computer, a pen computer, a microprocessor-based or programmable consumer electronics, a minicomputer, a mainframe computer, a personal mobile computing device, a mobile phone, a

portable or stationary personal computer, a palmtop computer or the like.

038 Processor **910** is, for example, a central processing unit (CPU), a micro-controller unit (MCU), digital signal processor (DSP), or the like.

039 Memory **920** symbolizes elements that temporarily or permanently store data and instructions. Although memory **920** is conveniently illustrated as part of computer **900**, memory function can also be implemented in network **990**, in computers **901/902** and in processor **910** themselves (e.g., cache, register), or elsewhere. Memory **920** can be a read only memory (ROM), a random access memory (RAM), or a memory with other access options. Memory **920** is physically implemented by computer-readable media, such as, for example: (a) magnetic media, like a hard disk, a floppy disk, or other magnetic disk, a tape, a cassette tape; (b) optical media, like optical disk (CD-ROM, digital versatile disk - DVD); (c) semiconductor media, like DRAM, SRAM, EPROM, EEPROM, memory stick, or by any other media, like paper.

040 Optionally, memory **920** is distributed across different media. Portions of memory **920** can be removable or non-removable. For reading from media and for writing in media, computer **900** uses devices well known in the art such as, for example, disk drives, tape drives.

041 Memory **920** stores support modules such as, for example, a basic input output system (BIOS), an operating system (OS), a program library, a compiler, an interpreter, and a text- processing tool. Support modules are commercially available and can be installed on computer **900** by those of skill in the

art. For simplicity, these modules are not illustrated.

042 CPP **100** comprises program instructions and - optionally - data that cause processor **910** to execute method steps of the present invention. Method steps are explained with more detail below. In other words, CPP **100** defines the operation of computer **900** and its interaction in network system **999**. For example and without the intention to be limiting, CPP **100** can be available as source code in any programming language, and as object code ("binary code") in a compiled form. Persons of skill in the art can use CPP **100** in connection with any of the above support modules (e.g., compiler, interpreter, operating system).

043 Although CPP **100** is illustrated as being stored in memory **920**, CPP **100** can be located elsewhere. CPP **100** can also be embodied in carrier **970**.

044 Carrier **970** is illustrated outside computer **900**. For communicating CPP **100** to computer **900**, carrier **970** is conveniently inserted into input device **940**. Carrier **970** is implemented as any computer readable medium, such as a medium largely explained above (cf. memory **920**). Generally, carrier **970** is an article of manufacture comprising a computer readable medium having computer readable program code means embodied therein for executing the method of the present invention. Further, program signal **980** can also embody computer program **100**. Signal **980** travels on network **990** to computer **900**.

045 Having described CPP **100**, program carrier **970**, and program signal **980** in connection with computer **900** is convenient. Optionally, program carrier **971/972** (not shown) and program signal **981/982** embody computer program product (CPP) **101/102** to be executed by

processor **911/912** (not shown) in computers **901/902**, respectively.

046 Input device **940** symbolizes a device that provides data and instructions for processing by computer **900**. For example, device **940** is a keyboard, a pointing device (e.g., mouse, trackball, cursor direction keys), microphone, joystick, game pad, or scanner. Although the examples are devices with human interaction, device **940** can also operate without human interaction, such as, a wireless receiver (e.g., with satellite dish or terrestrial antenna), a sensor (e.g., a thermometer), a counter (e.g., goods counter in a factory). Input device **940** can serve to read carrier **970**.

047 Output device **950** symbolizes a device that presents instructions and data that have been processed. For example, a monitor or a display, (cathode ray tube (CRT), flat panel display, liquid crystal display (LCD), speaker, printer, plotter, vibration alert device. Similar as above, output device **950** communicates with the user, but it can also communicate with further computers.

048 Input device **940** and output device **950** can be combined to a single device; any device **940** and **950** can be provided optional.

049 Bus **930** and network **990** provide logical and physical connections by conveying instruction and data signals. While connections inside computer **900** are conveniently referred to as "bus **930**", connections between computers **900-902** are referred to as "network **990**". Optionally, network **990** comprises gateways being computers that specialize in data transmission and protocol conversion.

050 Devices **940** and **950** are coupled to computer **900** by bus **930** (as illustrated) or by network **990** (optional). While the signals inside computer **900** are mostly electrical signals, the signals in network are electrical, magnetic, optical or wireless (radio) signals.

051 Networking environments (as network **990**) are commonplace in offices, enterprise-wide computer networks, intranets and the Internet (i.e. world wide web). The physical distance between a remote computer and computer **900** is not important. Network **990** can be a wired or a wireless network. To name a few network implementations, network **990** is, for example, a local area network (LAN), a wide area network (WAN), a public switched telephone network (PSTN); a Integrated Services Digital Network (ISDN), an infra-red (IR) link, a radio link, like Universal Mobile Telecommunications System (UMTS), Global System for Mobile Communication (GSM), Code Division Multiple Access (CDMA), or satellite link.

052 Transmission protocols and data formats are known, for example, as transmission control protocol/internet protocol (TCP/IP), hyper text transfer protocol (HTTP), secure HTTP, wireless application protocol, unique resource locator (URL), a unique resource identifier (URI), hyper text markup language HTML, extensible markup language (XML), extensible hyper text markup language (XHTML), wireless application markup language (WML), Standard Generalized Markup Language (SGML) etc.

053 Interfaces coupled between the elements are also well known in the art. For simplicity, interfaces are not illustrated. An interface can be, for example, a serial port interface, a parallel port interface, a

game port, a universal serial bus (USB) interface, an internal or external modem, a video adapter, or a sound card.

054 Computer and program are closely related. As used hereinafter, phrases, such as "the computer provides" and "the program provides", are convenient abbreviation to express actions by a computer that is controlled by a program.

Detailed Description of the Invention

055 For convenience, a list of references is provided prior to the claims.

056 **FIG. 2** illustrates a simplified block diagram of first computer **901** and second computer **902** that communicate according to the present invention. Computers **901** and **902** perform applications **201** and **202**, in first and second run-time environments, respectively.
Application **201** operates in a first run-time environment that has a first object model; application **202** operates in a second, different run-time environment that has a second object model. Objects **A1**, **B1** and **C1** in application **201** correspond to objects **A2**, **B2** and **C2** in application **202**. **FIG. 2** illustrates this mapping by dotted lines. For explanation, distinguishing the letters A, B and C provides sufficient object identification. Indices 1 and 2 conveniently distinguish the computers. Object **A2** is explained with more detail in **FIG. 5** (reference **152**).

057 In operation, computers **901** and **902** preferably visualize their performance on screens **951** and **952** to

first and second users, respectively. For example, screen **951** on computer **901** shows representations obtained from computer **902** (cf. **FIG. 4**, **118**, **128**, **138**). Both users are mentioned here only for convenience of explanation; the present invention does not require them to be involved.

058 Network **990** (cf. **FIG. 1**) is the infrastructure to convey information between computers **901** and **902** in both directions. In both computers, first and second run-time environments cooperate with each other, preferably, by sending messages through network **990**. The messages are convenient for explanation, but not essential for the present invention. In the alternative, the communication is performed by computer middleware, such as COM or CORBA, well known in the art and not further illustrated.

059 Communicator **101** operates on computer **901** as a message generator and sends messages GET **111**, SET **121**, INSTRUCT **131** to computer **902** (arrows to the right). As explained in connection with **FIG. 3**, the messages carry object identification and action identification.

060 Interpreter **102** (i.e. CPP) operates on computer **902** as message interpreter according to a method of the present invention (cf. **FIG. 4**). Interpreter **102** sends messages RESULT **112**, CONFIRM **122**, FEEDBACK **132** and ERROR **192** to computer **901**. Es explained in connection with **FIG. 4**, the messages carry object ID and response ID. It is an advantage of the present invention that the response ID has representations of object properties and object functions suitable for the first run-time environment of computer **901**. As mentioned, the representations can be shown on screen.

061 There is no need to use all messages that are illustrated in **FIG. 2**. Preferably, the messages are exchanged as paired messages (the first message from computer **901**, the second messages from computer **902**), each pair for an embodiment of the present invention (GET/RESULT, SET/CONFIRM and INSTRUCT/FEEDBACK).

062 The first embodiment relates to the action of reading (GET) a property of an object (A) und uses messages GET **111** and RESULT **112**. The second embodiment relates to the action of modifying (SET) a property of the object (A) and uses messages SET **121** and CONFIRM **122**. The third embodiment relates to the action of executing (INSTRUCT) a function of the object (A) and uses messages INSTRUCT **131** and FEEDBACK **132**. Error message **192** from computer **902** to computer **902** is optionally used in all embodiments.

063 An exemplary scenario relates to the automotive industry. A customer ("first user") uses first application **201** to customize a car; a technician ("second user") in a factory uses second application **202** to response to the customer. Software objects correspond to hardware components; objects **A**, **B** and **C** correspond to cars ALPHA, BETA, and GAMMA, respectively (not shown). Applications **201** and **202** communicate with their users in first ° and second °° natural languages (e.g., German and Spanish). To conveniently describe the invention in a single language, symbols ° and °° distinguish first and second natural languages, respectively. For example, "green °" stands for the German word "grün" and "green °°" stands for the Spanish word "verde", both in the same meaning of green color. In a real implementation, these superscripts are not shown.

064 For convenience of explanation, the example scenario further assumes consecutive execution of the embodiments. The customer for ALPHA

- first, asks for the color ("GREEN ° "),
- second, changes the color ("to RED °"), and
- third, communicates with applications **201** and **202** to check volume ("LITER °") and price ("EURO") for a desired color type ("METALLIC °").

065 **FIG. 3** illustrates messages GET **111**, SET **121** and INSTRUCT **131** from computer **901** to computer **902** (cf. **FIG. 2**, arrows to the right). The messages comprise object identification (ID) **115**, **125** or **135** (e.g., object A) and action identification (ID) **116/117**, **126/127**, or **136/137**. For first or second embodiments, the action ID comprises property ID **116** or **126** and request ID **117** or **127**. For the third embodiment, the action ID comprises function ID **136** and parameter ID **137** (parameter representation).

066 **FIG. 4** illustrates messages RESULT **112**, CONFIRM **122**, and FEEDBACK **132** from computer **902** to computer **901** (cf. **FIG. 2**, arrows to the left). The messages comprise object ID **115**, **125** or **135** (e.g., object A, as in **FIG. 3**) and response ID **116/118** or **126/128** or **136/138**.

For first or second embodiments, the response ID comprises property ID **116** or **126** (as in **FIG. 3**) and property representation **118** or **128**. Property representation is that of first run-time environment in computer **901**.

For the third embodiment, the response ID comprises function ID **136** (as in **FIG. 3**) and parameter representation **138**. Function and parameter

representation are suitable to first run-time environment of computer **901**.

067 The messages in FIGS. 3-4 are now explained in connection with the embodiments. So far, error messages **192** are neglected.

068 In the first embodiment, the customer needs to know the color of the car ALPHA. Application **201** uses object **A1** and communicator **101** to send GET **111** with object ID **115** "A", property ID **116** "COLOR", and request ID **117** "GET COLOR" (action).

069 At computer **902**, interpreter **102** executes a method (cf. **410** in **FIG. 8**) in combination with corresponding object **A2** (of application **202**) and returns RESULT **112** with corresponding object ID **115** and property ID **116**, but with property representation **118** "COLOR GREEN °" (response). Representation **118** is in the suitable format for application **201**. Application **201** now indicates "Green ° to the customer (e.g. on screen **951**).

070 The role of user of application **202** is optional. When executing the method, interpreter **102** converts representations to and from the different run-time environments. It is convenient (but not necessary) that application **202** shows the color to the technician in the second language °° or invites the technician to input the color in this second language °°.

071 In the second embodiment, the customer wishes to set the color of the car ALPHA to "red". Application **201** uses object **A1** and communicator **101** to send SET **121** to application **202** with object ID **125** "A", property ID **126** "COLOR", and request ID **127** "SET TO RED °".

072 At computer **902**, interpreter **102** executes a method (cf. **420** in **FIG. 8**) in combination with corresponding object **A2** (of application **202**) and returns CONFIRM **122** to application **201** with corresponding object ID **125** and property ID **126**, but with property representation **128** "YES, SET TO RED °".

073 In the third embodiment, the customer communicates with applications **201** and **202** to check price and volume for a desired color type. Application **201** again uses object **A1** and communicator **101** to send INSTRUCT **131** to application **202** with function ID **136** "CALCULATE" and parameter ID **137** "COLOR TYPE ° = METALLIC °", "VOLUME ° = X", "PRICE ° = Y". The parameters are input parameter (COLOR TYPE) and output parameters (VOLUME, PRICE).

074 At computer **902**, interpreter **102** executes the method in combination with corresponding object **A2**. Interpreter identifies object **A2**, verifies the parameters of **A2**, determines parameter representations, converts representations to the second-run time environment, triggers application **202** to execute the calculations, converts the calculation results X °° and Y °° to representations for the first run-time environment, and returns FEEDBACK **132** that indicates the output parameters (X °° and Y °°) as numeric values.

075 Before explaining the methods, details for the software at computer **902** are given.

076 **FIG. 5** illustrates object **152** (**A2**) in application **202** of second computer **902**. Object **A2** has property COLOR. COLOR has a plurality of property representations **162**

(RED °°, YELLOW °°, and GREEN °°) in the second run-time environment. In the example of **FIG. 5**, COLOR is represented by strings in the second natural language (°°). One representation of COLOR is being selected (e.g. GREEN °°). For the third embodiment, object **A2** also has function **172** CALCULATE with parameters COLOR TYPE, VOLUME, and PRICE.

077 **FIG. 6** illustrates conversion table **182** (mapping table) in application **202** (optionally in interpreter 102) on computer **902**. Due to different run-time environments in computers **901** and **902**, properties and functions of corresponding objects are represented differently. Table **182** illustrates property representations **161** and **162**. According to the present invention, interpreter **102** on computer **902** converts representations **161** from the format of the first run-time environment (RTE) to representations **162** (cf. **FIG. 5**) in the format of the second run-time environment and vice versa. As an example, table **182** converts string representations in the first language (°) of the property COLOR to a string representation in the second language (°°). Again, using natural languages is convenient for explanation; in practice there is great variety of conversion between different object models, such as integer-to-real number conversion; string-to-string conversion; one-byte character representation (e.g., ASCII) to double-byte representation (e.g., unicode), etc.

078 Persons of skill in the art can provide further property conversion means, for example: string-to-integer, integer-to-real, 4-byte-integer to 8-byte-integer, etc. and vice versa. Table **182** illustrates property representations only; function conversion

including parameter conversion is likewise to the property conversion, persons of skill in the art can accomplish this without the need of further explanation herein.

079 **FIG. 7** illustrates flowchart diagrams for method **400** of the present invention, with method **400** being a generalization for all embodiments.

080 **FIG. 8** illustrates flowchart diagrams for the embodiments: first method **410**, second method **420** and third method **430**.

081 Generally, method **400** relates to communication between first computer **901** operating in a first object-oriented run-time environment and second computer **902** operating in a second, different object-oriented run-time environment, wherein computer **901** sends message (GET **111**, SET **121**, or INSTRUCT **131**) with object ID **115**, **125**, **135** and action ID **116/117**, **126/127** or **136/137** to second computer **902** and causes second computer **902** to perform the method steps identify **401**, verify **402**, determine **403**, and execute **405**, **404**, **406**, **407**. The method steps are performed by interpreter **102** in combination with application **202** after computer **902** has received message **111**, **121**, or **131**.

082 In step identifying **401** an object in the second run-time environment according to object ID **115**, **125**, **135**, interpreter **102** identifies the corresponding object in application **202** by evaluating the object ID of the message. Step **401** is likewise for all embodiments, i.e. steps **411**, **421**, and **431**. In the

example scenario, with object ID "A", the object is **A2** (cf. **FIG. 5**).

083 In step verifying **402**, interpreter **102** verifies the existence of the action in the identified object in the second run-time environment. In the first and second embodiments, steps **412** and **422**, interpreter evaluates action identifier **116** and **126** and determines existence or non-existence of PROPERTY COLOR. In case of non-existence, interpreter **102** issues error message **192** (cf. **FIG. 2**). In the third embodiment, step **432**, interpreter evaluates action identifier **136** and determines existence or non-existence of function CALCULATE.

084 In step determining **403** representation **162** of action in the second run-time environment for the identified object, interpreter **102** operates for each embodiment slightly different. In the first and second embodiments, steps **413** and **423**, interpreter **102** determines that representations **162** are in string format (cf. **FIG. 5**, °°). In the third embodiment, step **433**, interpreter determines that the function has representation **172** (cf. **FIG. 5**).

085 In steps executing **404**, **405**, **406** and **407**, interpreter **102** executes the action by using representation **162** (or **172**). Interpreter **102** thereby operates for each embodiment in a slightly different manner. Common is obtaining the requested information (EXTRACT **414**, PERFORM **435**), converting formats (CONVERT **415**, CONVERT **424**, CONVERT **436**) and returning messages (RETURN **417**, **427**, **437**). Messages **112**, **122**, **132** to computer **901** comprise object ID **115**, **125**, **135** and response ID **116/118**, **126/128**, **136/138**. The following

description now conveniently splits into first, second and third embodiments.

086 **First Embodiment**

087 In step extracting **414**, interpreter **102** extracts representation **162** of property COLOR that is identified by action ID **116** (in GET **111**). So far representation **162** has the format for the second run-time environment (e.g., GREEN °°).

088 In step converting **415**, interpreter **102** converts representation **162** (GREEN °°) to a further representation (GREEN °, **161**) for the first run-time environment, for example, by reading from table **182** (cf. **FIG. 5**).

089 In step returning **417**, interpreter **102** return message RESULT **112** to the first computer with object ID **115** and response ID **116/118** (cf. **FIG. 4**). Response ID **116/118** comprises representations for the first run-time environment (GREEN °) as property representation **118**. In other words, interpreter **102** includes the extracted property into message **112** in a format suitable for the run-time of message-receiving computer **901**.

090 **Second Embodiment**

091 As property data is communicated to computer **902**, extracting and converting change the order, and inserting replaces extracting.

092 In step converting **424**, interpreter **102** converts request ID **127** (part of the action ID **126/127**) to representation **162** for the second run-time environment. For example, interpreter looking up in table **182** (cf. **FIG. 3**) to convert RED ° to RED °°.

093 In step inserting **425**, interpreter **102** inserts representation **162** into application **202**. In the example scenario, RED ° is stored in application such that car ALPHA will actually be painted red.

094 In step returning **427**, interpreter **102** sends message **122** to computer **901**. As explained above, CONFIRM **122** has object ID **125** and response ID **126/128** with property representation **128**.

095 **Third Embodiment**

096 Executing comprises converting **434**, performing **435**, converting **436**, and returning **437** steps. Function parameters are treated similar as properties.

097 In step converting **434**, interpreter **102** converts function ID **136** and parameter ID **137** of action ID **136/137** to function and parameter representations **172** (cf. **FIG. 5**) for the second run-time environment. For example, the function CALCULATE with its parameters COLOR TYPE (e.g. METALLIC), VOLUME (e.g., X) and PRICE (e.g., Y) is converted.

098 In step performing **435**, interpreter **102** causes the identified object of application **202** to actually provide the output parameters. The object uses function and parameter representations **172** for the second run-time environment. In the example, object **A2** calculates VOLUME by multiplying the surface area of the car (e.g., square meters) with an area specific volume (e.g., liters per square meter), object **A2** also calculates PRICE my multiplying VOLUME with a specific price for METALLIC (e.g., currency per liters). The parameter representations are still in the second run-time environment.

099 In step converting **436**, interpreter **102** converts the parameters (that result from performing step **435**)

into further representations **138** for the first run-time environment. For example, interpreter **102** converts VOLUME into a numeric representation suitable for the first run-time environment (e.g., integer to real number conversion; optionally conversion of physical units); interpreter **102** converts PRICE into a numeric representation suitable for the first run-time environment, conveniently with currency conversion.

0100 In step returning **437**, interpreter **102** returns feedback message **132** to computer **901** with object ID **135** and response ID **136/138**. Response ID **136/138** comprises the further representations for the first run-time environment.

0101 Optionally, function CALCULATE uses property COLOR (cf. first and second embodiments). This is convenient, for example, when COLOR influences output parameters such as PRICE.

List of Reference Numbers

Reference	Element	Embodiment
101	communicator	
102	interpreter (CPP)	
111	GET message	first
112	RESULT message	first
115	object ID	first
116	property ID	first
116/117	action ID	first
116/118	response ID	first
117	request ID	first
118	property representation	first
121	SET message	second
122	CONFIRM message	second
125	object ID	second
126	property ID	second
126/127	action ID	second
126/128	response ID	second
127	request ID	second
128	property representation	second
131	INSTRUCT message	third
132	FEEDBACK message	third
135	object ID	third
136	function ID	third
136/137	action ID	third
136/138	response ID	third
137	parameter ID	third
138	parameter representation	third
152	object A2	
161	object representation, first RTE	
162	object representation,	

Reference	Element	Embodiment
	second RTE	
172	function and parameter representation	third
182	conversion table	
192	ERROR message	all
201	application	
202	application	
400	method and steps	
401-407	method steps	all
41x	method and steps	first
42x	method and steps	second
43x	method and steps	third
901	computer	
902	computer	
951, 952	screens	
990	network	
A1, B1, C1	objects	
A2, B2, C2	objects	
ID	identification	
RTE	run-time environment	
xx1, xx2	indices to distinguish computers	

Claims

1. A method (400, 410, 420, 430) for communication between a first computer (901) operating in a first object-oriented run-time environment and a second computer (902) operating in a second, different object-oriented run-time environment, wherein the first computer (901) sends a first message (GET, SET, INSTRUCT) with object identification (115, 125, 135) and action identification (116/117, 126/127, 136/137) to the second computer (902) and causes the second computer (902) to perform the following steps:

   - identifying (401) an object (A2) in the second run-time environment according to the object identification (115, 125, 135);
   - verifying (402) the existence of the action (116/126/136) in the identified object (A2) in the second run-time environment;
   - determining (403) a representation (162/172) of the action (116/126/136) in the second run-time environment for the identified object (A2); and
   - executing (404, 405, 406, 407) the action by using the representation (162/172), thereby returning a second message (112, 122, 132) to the first computer (901) with object identification (115, 125, 135) and response identification (116/118, 126/128, 136/138).

2. The method (410) of claim 1, wherein step executing comprises:

- extracting (414) a representation (162) of a property (COLOR) that is identified by the action identification (116);
- converting (415) the representation (GREEN °°, 162) to a further representation (GREEN °, 161) for the first run-time environment; and
- returning (417) the second message as result message (112) to the first computer with object identification (115) and response identification (116, 118), the response identification (116, 118) with the representation for the first run-time environment (GREEN °).

3. The method (420) of claim 1, wherein step executing comprises:

- converting (424) a request identification (127) that is part of the action identification (126/127) to a representation (162, RED °°) for the second run-time environment; and
- inserting (425) the representation (162, RED °) into the second application (202).

4. The method (420) of claim 3, wherein step executing comprises to return (427) the second message as a confirmation message (122) to the first computer.

5. The method (420) of claim 3, wherein step converting (424) involves looking up in a table (182 RED ° to RED °°).

6. The method (430) of claim 3, wherein step executing comprises:

converting (434) function identification and parameter identification of the action identification (136/137) to function and parameter representations (172) for the second run-time environment;

performing (435) a function that is identified by the action identification by using the function and parameter representations (172) for the second run-time environment;

converting (436) parameters that result from executing into further representations (136, 138) for the first run-time environment; and

returning (437) the second message as a feedback message (132) to the first computer with object identification (135) and response identification (136, 138), the response identification (136, 138) with the further representations.

7. A computer program product (102) used in a communication system of a first computer (901) with a first object-oriented run-time environment and a second computer (902) with a second, different object-oriented run-time environment, wherein the first computer (901) sends a first message (GET, SET, INSTRUCT) with object identification (115, 125, 135) and action identification (116/117, 126/127, 136/137) to the second computer (902), the computer program product (102) embodied on a carrier (972) and having computer code instructions to cause a processor (922) of the second computer to interpret the first message, the instructions comprising:

code for identifying (401) an object (A2) in the second run-time environment according to the object identification (115, 125, 135);

code for verifying (402) the existence of the action (116/126/136) in the identified object (A2) in the second run-time environment;

code for determining (403) a representation (162/172) of the action (116/126/136) in the second run-time environment for the identified object (A2); and

code for executing (404, 405, 406, 407) the action by using the representation (162/172) and to return a second message (112, 122, 132) to the first computer (901) with object identification (115, 125, 135) and response identification (116/118, 126/128, 136/138).